Convex Optimization 10-725/36-725
Homework 3, due Oct 17

Instructions:

- You must complete Problems 1–3 and **either Problem 4 or Problem 5** (your choice between the two).
- When you submit the homework, upload a single PDF (e.g., produced by LaTeX, or scanned handwritten exercises) for the solution of each problem separately, to blackboard. You should your name at the top of each file, except for the first problem. **Your solution to Problem 1 (mastery set) should appear completely anonymous to a reader.**

1 Mastery set [25 points]

Be very concise. If written up and scanned, should be less than two sides in total for the whole mastery set. If Latex-ed up, should be less than one side.

Q1. [2+2+3] Project a vector $z \in \mathbb{R}^n$ onto

a) the non-negative orthant $\mathbb{R}^n_+$,

b) euclidean ball $\|x\|_2 \leq 1$,

c) ∞-norm ball $\|x\|_\infty \leq 1$

Q2. [3+3] Derive the prox operator $(\min_{x \in \mathbb{R}^n} (1/2)\|x - z\|_2^2 + \lambda h(x))$ for

a) $h(x) = \sum_i -\log(x_i)$

b) $h(x) = x^\top A x + b^\top x + c$ where A is a symmetric $n \times n$ psd matrix.

Q3. [4+4+4] Derive the duals of the following LPs

a) $\max_x 2x_1 + x_2$ subject to $x_1 - x_2 \leq 4, x_1 - x_2 \leq 2, x_1 \geq 0, x_2 \geq 0$

b) $\max_x 2x_1 + x_2$ subject to $-x_1 - x_2 \leq -4, x_1 + x_2 \leq 2, x_1 \geq 0, x_2 \geq 0$

c) $\max_x 2x_1 + x_2$ subject to $-x_1 + x_2 \leq -4, x_1 - x_2 \leq 2, x_1 \geq 0, x_2 \geq 0$

What can you say about the primal and dual feasibility and optimal values in all these settings?

2 Projections and proximal operators [25 points] (Adona)

Q1. [4+4] a) Project a symmetric real matrix onto the cone of all positive semidefinite matrices. b) Project a vector y onto the space $\{z : z = Ax + b, x \in \mathbb{R}^n\}$, for a given matrix $A \in \mathbb{R}^{m \times n}$ having full column rank

Q2. [5+7+5] Derive prox function for the regularizer ($h(x)$ function) a) $\lambda \sum_i (x_i)_+$ b), $\lambda \|x\|_2$, c) $\lambda \|x\|_\infty$.

Hint for part (c): Define

$$T_\rho(u) = \begin{cases} \rho \text{ if } u > \rho \\ u \text{ if } \rho \leq u \leq \rho \\ -\rho \text{ if } u < -\rho \end{cases}$$

Show that the solution is $x_i^* = T_\rho(z_i)$ where ρ is such that $\|x^* - z\|_1 = \lambda$.

3 Newton and quasi-Newton methods [25 points] (Sashank)

Let us revisit the functions given in Homework 1. In this assignment, we will compare the performance of few second-order methods on these functions.

$$f_Q(x, y) = 1.125x^2 + 0.5xy + 0.75y^2 + 2x + 2y \tag{1}$$
$$f_{LL}(x, y) = 0.5(x^2 + y^2) + 50 \cdot \log(1 + \exp(-0.5y)) + 50 \cdot \log(1 + \exp(0.2x)) \tag{2}$$
$$f_H(x, y) = 0.1(x^2 + y - 11)^2 + 0.1(x + y^2 - 7)^2 \tag{3}$$
$$f_R(x, y) = 0.002(1 - x)^2 + 0.2(y - x^2)^2 \tag{4}$$

This problem will involve programming. We recommend that you use Matlab or R. Still, you can use another language of your choosing (e.g., Python), but we will only provide support for Matlab and R.

(a) (6 pts) Implement Newton method with fixed step size 1. For every function, starting from origin, plot the trajectories on top of the contour of the function (like in Homework 1). The stopping criterion is $\|g_k\| < 10^{-6}$ where g_k is the gradient at point x_k.

(b) (10 pts) Implement BFGS line search from the same starting point. Recall, you need to implement line search for BFGS method. Use initial step size is 1, the back track ratio (β) is 0.5, the slope ratio (α) is 0.5, the maximum number of back tracks is 10. Use the same stopping criterion as above. Plot similar trajectories for BFGS method.

(c) (6 points) Compare your solutions (final points and function values) with Newton method. Are they same? If not, can you roughly explain the reason they converge to different solutions? Also, rerun the experiments and report the final points when you start from $(2, 2)$ and $(3, 3)$. Are they different?

(d) (3 pts) Now compare the number of iterations required for convergence in Newton, BFGS and gradient descent methods. Which one performs the best? It should be noted that computation required for each iteration is different for each of these methods. Compare the computational complexity of these methods for each iteration (give a rough count, in terms of an arbitrary dimension n for the optimization variable, even though we have $n = 2$ for this problem).

Along with your report please submit your code as well in a zip file to blackboard. Your implementation should be commented well enough to be able to understand, and it should also contain a batch file that can reproduce (up to a randomizing effect) all your plots.

4 Regularized logistic regression dual [25 points] (Yifei)

Consider the problem of regularized logistic regression

$$\min_{\beta \in \mathbb{R}^p} \sum_{i=1}^{n} -\Big(t_i \cdot x_i^T \beta - \log\big(1 + \exp(x_i^T \beta)\big)\Big) + \lambda \sum_{j=1}^{m} |d_j^T \beta|, \tag{5}$$

where $t_i \in \{0, 1\}$ are the observed outcomes and $x_i \in \mathbb{R}^p$ the predictors, $i = 1, \dots n$, and $d_j \in \mathbb{R}^p$, $j = 1, \dots m$ are given vectors. We'll refer to this as logistic regression with a generalized lasso penalty. Note that this generalizes the lasso logistic regression problem, which is a special case with $d_j = e_j$ (the jth standard basis vector) for $j = 1, \dots p$.

(a) (6 pts) Show that (5) can be equivalently written as

$$\min_{\beta \in \mathbb{R}^p} \sum_{i=1}^{n} \log\big(1 + \exp(-y_i \cdot x_i^T \beta)\big) + \lambda \|D\beta\|_1, \tag{6}$$

where $D \in \mathbb{R}^{m \times p}$ with jth row d_j and $y_i = 2t_i - 1 \in \{-1, 1\}$.

(b) (11 pts) When $X = I$ (i.e., $x_i = e_i$ for $i = 1, \dots n$), the above problem becomes

$$\min_{\beta \in \mathbb{R}^n} \sum_{i=1}^{n} \log\left(1 + \exp(-y_i \cdot \beta_i)\right) + \lambda \|D\beta\|_1. \tag{7}$$

Prove that the dual of the above problem is

$$\begin{aligned} \max_{\alpha \in \mathbb{R}^m} \quad & -\sum_{i=1}^{n} \Big(\big(y_i (D^T\alpha)_i\big) \log\big(y_i (D^T\alpha)_i\big) + \big(1 - y_i (D^T\alpha)_i\big) \log\big(1 - y_i (D^T\alpha)_i\big) \Big) \\ \text{subject to} \quad & -\lambda \le \alpha \le \lambda\,,\; 0 \le y_i (D^T\alpha)_i \le 1. \end{aligned} \tag{8}$$

[Hint: (7) can be written with slack variables as

$$\begin{aligned} \min_{\substack{\beta \in \mathbb{R}^n, \ell, u \in \mathbb{R}^m \\ \ell, u \ge 0}} \quad & \sum_{i=1}^{n} \log\left(1 + \exp(-y_i \cdot \beta_i)\right) + \lambda \sum_{j=1}^{m} (u_j + \ell_j). \\ s.t. \quad & w = D\beta,\; -\ell \le w \le u. \end{aligned}$$

]

(c) (2 pts) Among algorithms we've learned so far, would you be able to apply any to them to solve the primal problem (7)? Briefly explain why or why not. What about the dual problem (8)?

(d) (6 pts) What is the relationship between the solutions $\hat{\beta}$ and $\hat{\alpha}$ of (7) and (8)? If you knew a dual solution $\hat{\alpha}$, could you easily compute a primal solution $\hat{\beta}$? How about the other way around?

5 Nonnegative matrix factorization with gradient descent [25 points] (Yifei)

Let $\mathbf{V} \in \mathbb{R}^{m \times n}$ be a given matrix, and $0 < k < min(m, n)$ be a given positive integer.

The goal of nonnegative matrix factorization (NMF) is to solve

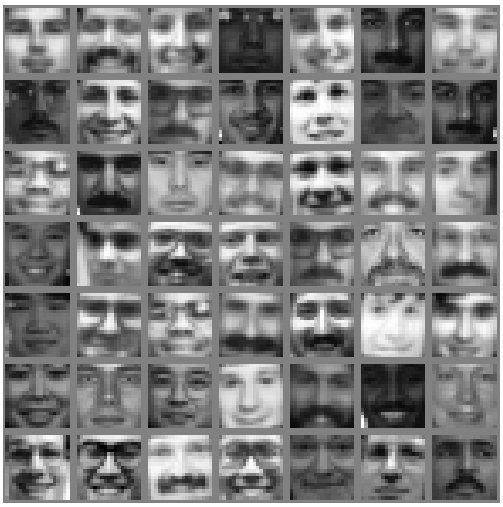
$$J = \min_{\mathbf{W} \ge 0, \mathbf{H} \ge 0} \|\mathbf{V} - \mathbf{W}\mathbf{H}\|_F^2 \tag{9}$$

where $\mathbf{W} \in \mathbb{R}^{m \times k}$, $\mathbf{H} \in \mathbb{R}^{k \times n}$, and $\mathbf{A} \ge 0$ denotes that $A_{ij} \ge 0$ for all i, j.

The NMF problem is a nonconvex problem, and therefore it is difficult to solve. However, if we fix $\mathbf{W}$, then the problem is convex in $\mathbf{H}$, and similarly if $\mathbf{H}$ is fixed, then the problem is convex in $\mathbf{W}$. Therefore a commonly used approach is to do alternating minimization: Fix $\mathbf{W}$, optimize J in $\mathbf{H}$, then fix $\mathbf{H}$ optimize J in $\mathbf{W}$ and repeat this process until convergence.

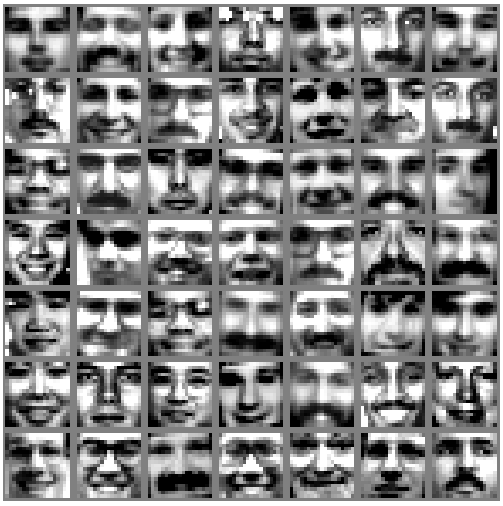
Interestingly, there is an elegant gradient descent algorithm (GDA) for this approach where the learning rate can be set such that GDA always generates feasible solutions, i.e. after the updates the $\mathbf{W}$ and $\mathbf{H}$ matrices remain nonnegative. The update rule is given in this paper Lee & Seung (2001) `http://hebb.mit.edu/people/seung/papers/nmfconverge.pdf` . More information about NMF can be found in `http://www.nature.com/nature/journal/v401/n6755/abs/401788a0.html` .

Your task is to implement this NMF algorithm on the CBCL FACE database `http://cbcl.mit.edu/software-datasets/FaceData2.html` . A sample of the original faces look like

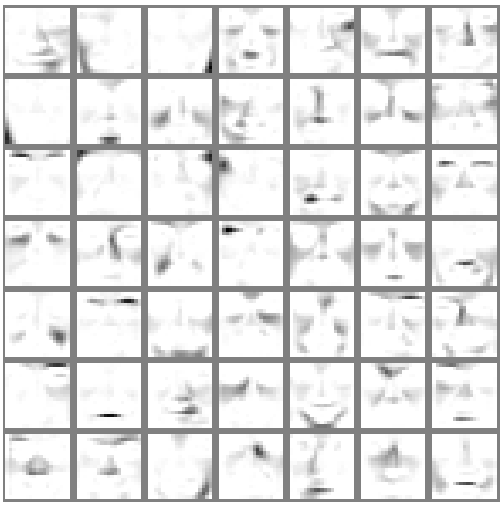


(a. 5 points) Take the original faces from face.txt , apply the following preprocessing:

For every face that is stored in a column of $\mathbf{V}_i$, normalize it such that the median $\mathbf{m}_i = 0.5$ and the median of the deviation from the median is 0.25, i.e. $\underset{j}{median}(|\mathbf{V}_{ji} - \mathbf{m}_i|)$=0.25. Truncate values above 1 or below $1e-4$ (because we do not want exact zeros either). Now, we have a $\mathbf{V}$ that looks like (sampled columns):



(b. 10 points) Apply the algorithm Eq(4) in the paper Lee&Seung (2001) to optimize (9). $k = 49$. Initialize every entry in your $\mathbf{W}$ and $\mathbf{H}$ as a random draw from uniform(0,1). Run 300 iterations. Plot the columns of $\mathbf{W}$ similar to the bellow. Also plot the objective value in every iteration. What columns of $\mathbf{W}$ are more important?



(c. 5 points) Show, following the arguments in Lee & Seung (2001), that the algorithm you have implemented is indeed performing gradient descent steps, with a cleverly chosen step size that ensures the matrices W, H will have nonnegative entries at each step. What is an alternative, still using gradient descent, to ensure that the entries of W, H are nonnegative?

(d. 5 points) Now load face_test.txt , reconstruct the faces by optimizing $\mathbf{H}$ with your learned $\mathbf{W}$ fixed. What is the mean mean-squared-error of each reconstructed images (the objective value normalized by the number of images)? Do your reconstructed faces look similar to test faces?

To submit for (b) and (d) Submit the plots for parts (b) and (d) to answer the questions asked. Please also submit the following functions (or similar functions if you are using another language) in a zip file.

(a) `function [V_normalized] = normalize_faces(V)`
(b) `function [W, H, objective_val_history] = nmf(V_normalized, k)`
(c) `function [objective_val_history, H] = test_faces(W, testV_normalized)`

[Hint: the function compact_canvas.m may come handy to plot the faces. You may find $\mathbf{V}$ and $\mathbf{V}_{test}$ in face.txt and face_test.txt]